jamk.fi

Implementing Ethereum blockchain to RGCE cyber range

Aleksi Ahola

Bachelor's thesis
May 2020
Technology
Degree Programme in Information Technology
Cyber Security

jamk.fi

Description

Author(s)	Type of publication	Date
Ahola, Aleksi	Bachelor's thesis	May 2020
		Language of publication: English
	Number of pages: 50	Permission for web publication: Yes

Title of publication

Implementing Ethereum blockchain to RGCE cyber range

Degree programme

Information Technology

Supervisor(s)

Saharinen Karo

Assigned by

JYVSECTEC, Vatanen Marko

Abstract

New technologies are emerging at a fast pace and many of them fail in the beginning. Some of new technologies manage to stick around and become more successful. Blockchain has been around for almost a decade and it has shown great promise in many different fields. New technologies create a need for research, as they cannot be blindly trusted when not everything is known about them. Research was done on the blockchain technology and Ethereum blockchain, focusing on the security aspect. The aim was to find out if a blockchain network could be useful in the RGCE Cyber Range as part of research and training.

The outcome of the research was a better understanding of the technology behind blockchain and how secure different parts of it are. To increase readiness and understanding of possible attacks, research on different attack vectors on blockchains and Ethereum was conducted. Implementing a private Ethereum blockchain in the RGCE Cyber Range will allow the JYVSECTEC to research and possibly use the decentralized network in their cyber trainings.

During the research it became clear that even though blockchains are considered very safe technology, they have vulnerabilities that need to be considered when using them. A blockchain network is safer when compared to traditional centralized networks, but it often has applications built on top of the network, which pose a risk of potential attacks against it. Every part built on top of the blockchain network increases the amount of attack vectors.

Blockchain just as every other technology has its weaknesses but using following best practices for everyday IT such as password is sufficient for everyday users.

Keywords/tags (subjects)

Blockchains, Ethereum, Networks, Cyber Security,

Miscellaneous (Confidential information)

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ahola, Aleksi	Opinnäytetyö, AMK	Toukokuu 2020
		Julkaisun kieli Englanti
	Sivumäärä 50	Verkkojulkaisulupa myönnetty: Kyllä

Työn nimi

Implementing Ethereum blockchain on RGCE cyber range

Tutkinto-ohjelma

Tieto- ja viestintätekniikka

Työn ohjaaja(t)

Karo Saharinen

Toimeksiantaja(t)

JYVSECTEC, Marko Vatanen

Tiivistelmä

Uusia teknologioita ilmestyy nopealla tahdilla ja monet niistä epäonnistuvat jo alkumetreillä, mutta jotkin niistä onnistuvat pysymään suosittuina ja menestymään. Lohkoketjut ovat olleet olemassa yli 10 vuotta, ja ne ovat näyttäneet kykynsä taipua moneen eri käyttötarkoitukseen. Uudet teknologiat luovat tarpeen niiden tutkimiselle, sillä uusiin teknologioihin ei voida sokeasti luottaa, kun kaikkea niistä ei vielä tiedetä. Tutkimus lohkoketjuista ja Ethereum lohkoketjusta tehtiin keskittyen turvallisuuteen. Tavoite oli selvittää, että onko lohkoketjusta hyötyä osana RGCE ympäristöä tutkimusta ja mahdollisesti kyberharjoituksia varten.

Tutkimuksen tuloksena saavutettiin parempi ymmärrys lohkoketju teknologiasta ja sen eri osa-alueiden turvallisuudesta. Hyökkäysvektoreiden ja -rajapintojen tutkiminen lisää ymmärrystä ja valmiutta hyökkäyksien tapahtuessa. Implementoimalla yksityisen Ethereum lohkoketjun RGCE ympäristöön mahdollistetaan JYVSECTEC:in hyödyntävän sitä tutkimuksissa ja mahdollisesti kyberharjoituksissa.

Tutkimuksen aikana selkeni, että vaikka lohkoketjuja pidetään turvallisena teknologiana, niin täysin turvallisia ne eivät ole. Lohkoketjujen haavoittuvuudet tulee huomioida niitä käyttäessä. Lohkoketju verkko on turvallisempi verrattuna perinteisiin keskittyihin/centralized verkkoihin, mutta lohkoketjujen päällä on usein aplikaatioita, jotka lisäävät hyökkäysten mahdollisuutta. Jokainen blockchainin päälle lisätty ominaisuus tai aplikaatio lisää hyökkäysvektoreita ja joitakin osia voidaan pitää epäturvallisina, jos niiden käyttäjät ja kehittäjät eivät ole tietoisia heikkouksista.

Lohkoketjuilla aivan kuten muillakin teknoloigoilla on omat heikkoutensa, mutta yleisiä turvallisuuskäytäntöjä noudattamalla saadaan mitigoitua suurin osa yleisistä alttiuksista.

Avainsanat (asiasanat)

Lohkoketjut, Ethereum, Kyberturvallisuus, Verkot

Contents

Figures

Acronyms

API	Application Programming Interface
CA	Certificate authority
DAG	Directed Acyclic Graph
Geth	Go-Ethereum
Eth	Ethereum
EIP	Ethereum Improvement Proposal
JS	JavaScript
RLP	Recursive Lenght Prefix
BFT	Bysantine fault tolerance
DApp	Decentralized Application
P2P	Peer-to-peer
ASIC	Application-Specific Integrated Circuit
CPU	Central Processing Unit
GPU	Graphical Processing Unit
EVM	Ethereum Virtual Machine
PoW	Proof of Work
PoS	Proof of Stake
DDoS	Distributed denial of service
RPC	Remote Procedure Call
DAO	Decentralized Autonomous Organization
UI	User Interface
LAN	Local Area Network

1 Introduction

Blockchain and the cryptocurrencies have been gaining popularity, and with that so has been scams and attacks against them. Proponents of blockchain consider it to be the one of the best ways to create transactions securely (Miles, 2017). Considering a technology as the best at anything should always be taken skeptically as technologies are constantly evolving and blockchain is no exception. Attacks on wallets and 51% attacks on networks are possible and this needs to be considered when using blockchains. Blockchain technology is also vulnerable to DDoS attacks, even though DDoS attacks against a blockchain is harder than attacking a normal network as the workload is shared between users of the network. Smart contracts are new and experimental part of blockchain networks, and they can be used in a wide variety of ways including DApps. Everything build on top of blockchain is another attack vector for hackers and malicious actors.

1.1 Motivation

In the rapidly growing and developing environment of technologies it is crucial to keep up with new technologies. Cyber security requires keeping up with technologies and their weaknesses. Blockchain is a relatively newly adopted technology that has been growing in the past years rapidly. Every aspect is still not known about blockchains and research is ongoing both from the developers and the malicious actors. While new technologies emerge, they should be considered and researched from a cybersecurity perspective to keep Internet safe. Research and training with new technologies is an undisputed necessity.

The goal of training and research about blockchain is to increase knowledge and preparedness about the possibilities that can occur both beneficial and malicious. Understanding the technology increases the possibilities it can be used in both research and development. Recognizing possible vulnerabilities of different parts of blockchain is required to effectively being able to prevent attacks and other malicious actions. By understanding the technology better, the skills to prevent and react to different scenarios with the technology and other parts of it are increased.

Implementing a blockchain in an RGCE will enable training and understanding what is required to build a decentralized network. Ethereum is the chosen blockchain, as it supports private networks, meaning it can be used in a closed environment. Another motive to use Ethereum is smart contracts, which have shown great promise with trustless transaction contracts. As smart contracts are an experimental part of blockchains, it requires understanding about the possible vulnerabilities to counteract them. Experimental technologies such as smart contracts are updated and developed continuously, but so are the attacks on them, as they are still new, not every aspect and vulnerability are known about them.

The blockchain can resist traditional attacks well, but cybercriminals are developing new attacks specifically for hacking blockchains and other parts of it. All technology has its attack vectors and blockchains are no exception.

1.2 Research Frame of this work

Implementing an Ethereum blockchain in the RGCE cyber range platform aims to answer to a question. Will a blockchain network be useful for future research and training purposes. By implementing a blockchain environment comparable to the real blockchain in the Internet can we know if it will be useful or not. After the implementation research on the security of blockchains will increase knowledge about the possible weaknesses. Researching blockchain hardening and other mitigations will be the secondary purpose of the thesis.

Priority in the RGCE-network is being realistic, meaning the private Ethereum should be used as close as Ethereum in the Internet would be. Other consideration is usability, since Ethereum runs as a node it needs to be controlled using various applications. Without tools, such as wallets that can be used to make accounts and transactions everything would be done from JS command line, each action requiring a new command or line of commands. For the research and training purposes the usability is very important so researchers and training participants will not have to learn all the commands required to use the blockchain, but instead interact with the blockchain by using an UI.

1.3 Research methods

The research started by reading extensively about different parts of blockchains and Ethereum after which research about the security of the blockchains and different attack vectors on Ethereum was conducted. Security and attack vectors were researched and analyzed to understand the vulnerabilities thoroughly. It is important to understand the different vulnerabilities of each part of blockchains to use it more safely.

For the implementation, the research included learning about private networks and different applications used for the created blockchain environment in the RGCE.

2 Ethereum

"Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference" (*Ethereum*). Ethereum was originally proposed at the end of 2013 by Vitalik Buterin. In 2014 Ethereum collected funding by crowdfunding, and the collected money was used for development and publishing of the Ethereum software. Ethereum has its own cryptocurrency called Ether or Eth, used for making transactions and as a currency.

Before the Ethereum went live, 60 million Eth was created. 80% of the 60 million Eth was available for purchase, and the other 20% was retained for the early contributors and the developers of Ethereum (EthHub).

2.1 Technology behind Blockchain

"Blockchain is a digital ledger of economic transactions that is fully public, continually updated by countless users, and considered by many impossible to corrupt" (Carlazo, 2017). Each block is encrypted by hash of the previous block effectively chaining the blocks. Every block has the hash of previous block embedded in the block data, which means that blocks can not be edited after they have been made since the hash of next block would not match anymore because of the changes made.

2.2 Brief History of Blockchain

Satoshi Nakamoto published a paper about Bitcoin in 2008 and in the following year Bitcoin software was published. These events are considered the birth of blockchain. Blockchain is not a new invention. Decentralized and cryptographically secured data chains and currencies have been proposed before without a proper success. Stuart Haber and W. Scott Stornetta published a paper called “How to time-stamp a digital document” proposing cryptographically secured chain of data in the year 1991 (Haber, Stornetta, 1991). They tried to create a system where timestamps of documents would become immutable by using the merkle tree design, which is a tree of cryptographic hashing. Their design also included consensus algorithms to confirm data. In Satoshi Nakamoto’s Bitcoin paper, Haber and Stornetta are cited multiple times.

2.3 Blocks

Blockchain consists of blocks that have been chained to previous and to the next one by using hash from the previous block’s header. Each block contains certain data fields, transactions and extra data if needed. (See Figure 1.).

The genesis block is the first block that begins the chain. Since the first block does not have any reference to previous blocks it has to be handwritten and hardcoded in the configurations. Genesis block contains the configurations for the blockchain as parameters that are in a .json file that can be used to initialize the genesis block.

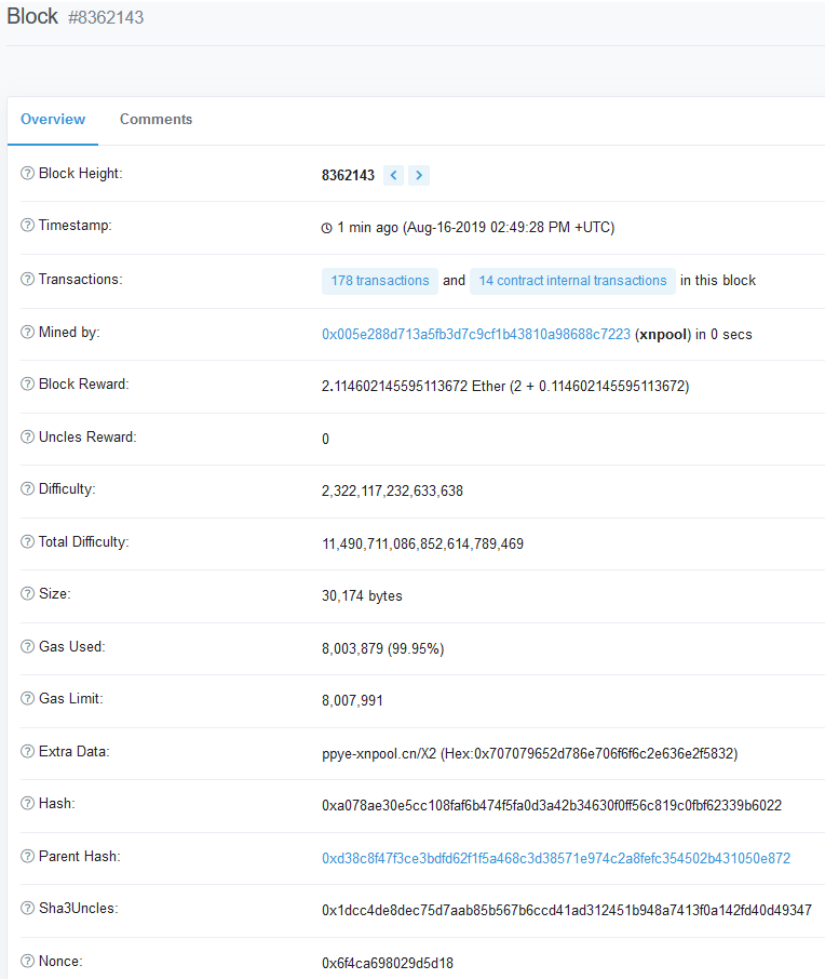

Figure 1 Example of Ethereum block (Etherscan)

Block Height also known as Block Number, an ordinal number of the current block indicating the length of the blockchain.

- **Timestamp** is the time of block being mined in Unix time() format. Time is used for calculating mining difficulty, if time between blocks gets too short, the difficulty rises and if the time between blocks is too long, the difficulty decreases.
- **Transactions** contains all the transaction data and contract internal transactions.
- **Mined by** indicates the address and the possible pool of miner who has successfully added the block to the blockchain.
- **Block reward** is the reward for mining the block. Reward size keeps changing slightly, but the basic reward is 2 Eth + fees paid for transactions in the block what amounts around 0,01 to 0,2 Eth, depending on the gas spent for transactions.
- **Uncle reward** indicates if an uncle block has been mined successfully. Uncle blocks happen when two miners generate a block simultaneously and other one gets rejected. Uncle block has a significantly lower reward than a normal block. Uncles reward is valid but rejected as it is not on the longest chain which is the working mechanism of the blockchain. Uncle block secures the blockchain. (Etherscan).
- **Difficulty** indicates the amount of effort required to mine the block. Normally difficulty is automatically calculated from previous blocks difficulty level and timestamp. Genesis block does not have previous blocks so the difficulty has to be set manually. Generally it is good idea to start with low difficulty, for the first blocks to be mined quickly without requiring huge amounts of computational power.

- **Total difficulty** indicates the total difficulty of the chain until the current block. Total difficulty is calculated by summing all difficulties of previous blocks.
- **Size** indicates the amount of data the block takes. Increasing the gas limit increases the block's size. Average sizes are between 1 000 and 50 000 bytes.
- **Gas used** in the block and percentage of gas used in the block and the GasLimit is a value that restricts gas usage per transaction. Computational work required to run transactions and smart contracts is measured by Gas (Zainuddin). Gas is a small amount of Eth comparable of about 0,02€ payed for transferring Eth. Transaction maker can pay more Gas to make transaction go faster and going in front of the line of transactions.
- **Extra data** is any data that the miner of the block can include in the block
- **Hash** is the hash of the block header from the current block and parent hash is the hash of the previous block. Keccak-256 is the algorithm used for hashing in Ethereum blocks to chain them.
- **Sha3Uncles** is a hash combined from all uncles before the block. If no uncles are found then the Sha3Uncles stays the same.
- **Block Nonce** is a 64-bit string hash, which is used for hashing. Essentially it is a value used in mining to demonstrate enough work was required to mine the block.

Of these values Genesis block needs *timestamp*, *gaslimit* and *difficulty* to be hard-coded. In addition, Genesis block also takes additional values and configurations.

Alloc is a set of two parameters which are used to allocate Eth to specified addresses. First parameter is the address Eth is allocated to and the second parameter is amount of Eth allocated in Wei, which is the smallest unit of Eth. One Ether equals to 1e18 Wei.

Genesis block also contains the configuration for the blockchain network that are only in the genesis block. (See Figure 2.)

ChainID is an integer which is used to identify different blockchain networks, as many blockchain networks can be ran at the same time. Main Ethereum ChainID is 1.

HomesteadBlock is used to specify the version of Ethereum used; 0 = use homestead, nil = use old Geth version.

EIP150Block is used to increase gas price, to prevent DoS-attacks. *EIP155Block* is used to prevent replay attacks. *EIP158Block* treats empty accounts as non-existent to make the data amount of blockchain smaller. EIPs are new features that are used in the main network but are optional in the private network.

Figure 2 Genesis block file

2.4 Byzantine fault tolerance

Blockchain networks consists of nodes, which all act on information available for them from other nodes. Reaching consensus between nodes is required to include transactions into the blocks mined. Problem of reaching consensus is malicious actors, and the false information they might provide to other nodes. To solve the problem of malicious node in the pool of regular nodes, modified Byzantine fault tolerance is implemented in blockchain in the form of PoW as the consensus algorithm. It is used to define failed actions and to reach secure trustless consensus.

Every decentralized network uses a form of Byzantine fault tolerance. Byzantine fault tolerance comes as a solution to problem conceived in 1982 as the Byzantine Generals' problem. The dilemma assumes that each general has its own army and that each group is situated in different locations around the city they intend to attack. The generals need to agree on either attacking or retreating. "It does not matter whether they attack or retreat, as long as all generals reach consensus, i.e., agree on a common decision in order to execute it in coordination" (Binance Academy). Byzantine fault tolerance uses nodes as "generals" in the problem.

Each general decides to act separately. Action is to attack or to retreat, or in the context of blockchain yes or no. After decision is made it is final and they can not change their decisions. In order for attack or retreat to work perfectly, every general has to reach consensus.

This works as a base for Byzantine Generals' problem, but the problem comes when generals can only communicate via a message courier. The main problem of this in the context of blockchain is when one of the generals says the opposite of what he was told or disagrees with others just to cause confusion and possibly failure of others.

Solution to the Byzantine Generals' problem is simple. If 2/3 or more of the generals must agree in order to reach consensus. This also means if 2/3 or more of the generals are corrupt and have chosen to do malicious actions, it will happen, and this is exactly what makes 51% attack possible.

At the moment Ethereum PoW is the chosen solution to Byzantine generals' problem, but later PoS will overtake PoW's position as it does not consume electricity nor require massive amounts of calculating power to use.

2.5 Ethash

Ethash is the PoW algorithm used in Ethereum which is based on the original Dagger Hashimoto mining algorithm. One of the most important things for a good hashing algorithm is its ability to resist ASICs. ASIC resistance makes the network more equal to miners and it can prevent from 51% attacks. Dagger Hashimoto, which was a predecessor of Ethash was an ASIC resistant algorithm, meaning that the benefits of building and developing an ASICs would have been minimal. ASIC resistancy ensures that running the algorithm on a normal computer CPU or GPU would still be profitable for the users. Ethereum is traditionally mined using GPUs which are meant to render graphics for computers, but they are very efficient at calculating complex mathematical problems, such as Ethash.

Ethereum has been resisting ASIC mining since its starting days, but in 2018 an ASIC specifically designed for Ethereum was released. Ethereum specific ASICs are more efficient at processing Ethash compared to GPUs. ASIC mining poses a risk of huge mining farms controlling a great percentage of Ethereum's hashrate. There has been a race of sorts between ASICS developers and different blockchains, ASICs developers create a new machine to circumvent ASIC resiliency and blockchains updates the

function to be more ASIC resilient. If a very powerful and efficient ASIC was developed, it could be used to attack the blockchain.

A DAG is a large randomly generated dataset used in the Ethash algorithm to make it more ASIC resistant (Ethereum GitHub). A new DAG is generated for every epoch, which is set of 30 000 blocks.

In the July of 2020 Ethereum is updating the hashing algorithm with Programmatic Proof of Work or ProgPow. The ProgPow fork follows the EIP-1962 update in the June of 2020, which contains small changes to the cryptographic functions of Ethereum. ProgPow aims to decrease the ASIC advantage over GPUs to a maximum of 20%. People opposing this update might cause a hardfork, meaning that there would be two different chains running parallel, one using ProgPow and other using the old PoW algorithm.

2.6 Ether

Eth is the currency of Ethereum and it is made by mining blocks and before the Ethereum main net went live, donators were given some as an exchange for money. Mining a block has static 2 Eth reward + fees that were used for transactions in that block. A miner that successfully mines the block will get the reward to their account. When new block is mined, previously saved data of users is now locked putting them in to the block and the Eth is rewarded. In order for users to have Eth as currency they need an Ethereum wallet or in Geth an account to store Eth. In addition to the wallet's normal functions, some Ethereum wallets are used to interacting with smart contracts and make transactions.

2.7 Security

Blockchain is far more less vulnerable compared to traditional networks and storage of data. Uneditable ledgers and data distribution across P2P network makes sure that data cannot be edited or tampered with.

Ethereum just as every other blockchain is considered trustless. Being trustless means two entities can securely perform transactions without knowing and trusting each other. This is achieved by having a third entity between the two who are doing the transaction, to do the transaction and the blockchain does just that. Blockchain will always perform the transaction just as it is told to do, with use of PoW. Consensus protocol is a key factor in trustless transactions. Without a consensus protocol, anybody could tamper the transaction.

Even though blockchain networks are considered secure, infrastructure behind the blockchain defines the total security. If infrastructure is vulnerable to attacks and access, the blockchain can be tampered with and great damage can be done.

Not only blockchains, but operating systems and and the wallets where the currency is stored are attack vectors. Building more functionality on top of the blockchain means more attack vectors for malicious actors.

2.8 Ethereum 2.0

Major limitation of Ethereum is scalability, Ethereum has shown increased transaction times and higher fees, during high user activity. There is a new version of Ethereum being developed by Vitalik Buterin, the writer of the original Ethereum whitepaper. He has revealed that Ethereum 2.0 is being developed and first phase will be launched in 2020. Ethereum 2.0 will be PoS network, with that also making 51% attacks more unlikely to happen.

CASPER is the PoS protocol Ethereum has chosen and is moving towards in the Ethereum 2.0. Casper protocol works by validators staking Eth as stake and after that they start to validate blocks. Validators discover blocks which they think can be added to the blockchain and validation happens by placing a bet on block. If a block is added to the chain, the validators get reward based on the bet they have placed on the block. Casper is more Byzantine Fault Tolerant and aims to be a trustless system. Casper is not a single project, but instead a combination of two projects which are worked by the Ethereum development team. The two projects are: Casper Finality Gadget, which is a hybrid PoW/PoS consensus meant to ease the transition to purely PoS system, and Casper the Friendly GHOST: correct-by-construction.

CBC is very abstract concept of building new protocols that are used to build better new protocols. New protocols are built by incrementally refining the previous successful protocols.

“It is called correct-by-construction because correctness of new protocols is guaranteed by their construction, which is by incrementally defining an abstract (and provably correct) protocol further” (Asgaonkar, 2018.).

2.9 Smart Contracts

Smart contracts are essentially software programs that are using blockchain’s distributed data storage (Praitheeshan, Pan, Yu, Liu & Doss). While originally blockchain was mainly used as a ledger or secure way of storing data, the DApp development has gaining popularity and smart contracts are the essential part in DApps. Smart Contracts are used in applications as autonomous entities acting as the middle man. Smart Contracts can be used simply as smart contracts, where they verify the negotiation of a contract without third parties (Rosic 2017).

DApp is an application that runs on blockchain instead of a computer. DApp is a term specifically used for blockchain applications, but there are other decentralized applications like torrents. Torrents run on a P2P network, instead of blockchain.

Popular way of comparing DApps to traditional websites or web applications is that instead of API there is smart contract and instead of database there is a blockchain. Backend is ran from decentralized blockchain network and it works as a database and smart contracts are used to run transactions on it.

2.9.1 Smart Contract Usage

Smart Contracts are mostly used for exchanging anything of value in very easy and conflict-free way without having to pay to the middleman. Basically smart contracts are protocols of transactions. Smart Contracts are like vending machines, you give money and get something in exchange, very straight forward and no need for other

entities besides the customer and the machine itself. In terms of Ethereum you an give amount of Eth and get service back.

Smart contracts have terms of agreement, which the transactions rely on. If the terms are not met the transaction will not be executed. Terms are written directly into the code of smart contract. Smart contracts hold currency to make transactions and pay for gas. Currency held by the contract is can vary a lot. More often the contract is used the more currency it needs to complete these transactions. Many smart contracts hold significant amounts of Eth, which makes them desirable targets for attackers.

2.9.2 Solidity

Smart contracts are written in Solidity. Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs that control the accounts in the blockchain (Solidity). Solidity enables creation of contracts including multiple people, such as voting and blind auctions. Solidity is still in beta 0.x version, and beta products tend to change a lot. Solidity is updated regularly and that can cause old code in contracts to break, but regularly updating the Solidity means less bugs and vulnerabilities.

3 Innovative uses of blockchain

3.1 Usecase: Estonian health records

In 2016 Estonia was looking for a new more secure ways for securing the health records and were looking for new and innovative technologies. During the search multiple cryptocurrency developers and investors were looking big implementations to demonstrate the power of blockchain technologies. Securing the health records of Estonia by using blockchain was decided on and Estonia became the first country to use blockchain as a nation.

After concluding on blockchain for securing the health records, Estonian E-Health Foundation launched the development for blockchain that would later be known as KSI blockchain.

The KSI blockchain is used to produce an additional layer of security to help maintain health record integrity. KSI blockchain is not used to secure the health records directly, but instead to store files that basically contain data-access logging of the health records. By securing the data-access logs with blockchain an unchangeable audit trail is created, auditing said blockchain is proven to be secure as the data cannot be changed or tampered with afterwards.

3.2 Usecase: Internet of Trusted Things

Internet of Trusted Things is a new term used for IoT integrated with a blockchain. According to Dongxing, Wei, Wenping & Fangyu (2018, 6) blockchain is looking to solve problems of centralized security models and infrastructures such as DDoS, single point failures and scaling infrastructure.

Generally if IoT is using blockchain, the IoT devices work as nodes connected to the blockchain network. Blockchain can be used as the database for IoT network, but as the network grows, so does the data that has to be processed, more data requires more computing power which makes scaling a problem for IoT devices. As the strong cryptographic algorithms produce latency, and latencies are not suitable for real time IoT devices.

Most promising feature for blockchain to fix in the IoT field is the authentication. Identity authentication system for IoT devices is dependant on a CA server, which is at risk of single-point-failure attack (Dongxing, Wei, Wenping & Fangyu, 2018, 2).

In the undregulated IoT market, the security of devices is a problem that enables device hacking (Compton, 2017). IoT is at risk of hacking as much as any other computer which is poorly protected if using insecure components or outdated software.

One of the biggest problems with blockchain in the IoT devices is the power required to run PoW mining. Devices need to be powerful enough to process the blockchain actions and even if they are powerful enough, the 51% attack is still an attack vector.

Protection against 51% and less power consuming more efficient PoW or PoS algorithms need to be developed to secure the IoT market and to make blockchain more viable part of IoT.

3.3 Usecase: RGCE

Platform for the Ethereum blockchain is RGCE (Realistic Global Cyber Environment). RGCE is owned and ran by JYVSECTEC at Jyväskylän ammattikorkeakoulu in Jyväskylä. RGCE is multipurpose environment for cyber security training and exercises. RGCE is used as a realistic cyber environment, meaning it is built to perform like Internet, real machines, traffic and applications would. RGCE is used for research and development, and for training public authorities.

Geth provides a real Ethereum network, that works basically same as the main network. It is run as private network, meaning it can be controlled and altered by JYVSECTEC engineers. It can be used to create more depth in the realistic environment and allow people interact with blockchain as they would in the Internet.

Ethereum could be used for ransomware campaigns, as means of paying the malicious actor in a campaigns or as asset of blue team, which could be attacked. Smart contracts can be implemented for training or research for smart contract based attacks. Smart contracts also enable developing DApps for the environment, if they are seen as a beneficial part of training and research.

4 Implementation

Implementation from empty machine to run blockchain, either an Internet connection must be established, or the needed repositories have to be downloaded in to the private network prehand. The blockchain is run as a decentralized network, meaning after everything required to run the blockchain by starting the Geth, the connection to the Internet may be cut completely. Geth supports private networks and they do not require Internet connection to be run. Although Internet is not

needed to build a blockchain network, a working connection between nodes have to be established in order to transfer data i.e. local network. All the data is moving in the private decentralized network created by the blockchain software.

4.1 Go-Ethereum

Geth is the official implementation of the Ethereum used to implement the blockchain and create private blockchain network. Implementation platform of choosing is Ubuntu, but Geth can be run from Windows, most common Linux distributions and MacOS. Installation manual for Geth is found in Appendix 1.

As one of the goals of this thesis is to make Ethereum as easy to use as possible, avoiding the need to use JS command line is important for the ease of usage. Geth can be used as a standalone Ethereum platform. Everything can be done from JS command line but using a wallet application makes the using and accessing the blockchain much easier. Used applications allow using and monitoring the blockchain take a lot less effort. For usage in cyber security training the applications are essential, so the participants do not have to learn how to use Geth from JS command line. Using applications with graphical interface is more intuitive and in most cases more realistic.

Ethereum mining can be done in the background from one or couple of computers doing all the mining needed. Ease of mining has been taken into consideration and it does not require any notable amounts of energy or computing power to mine blocks when difficulty is set to a low value.

4.2 Installation

Installing private blockchain in to the RGCE cyber range is done as it would be done on regular computer connected to the Internet as the blockchain does not connect to the Internet, but instead creates a decentralized network itself. The Geth is installed and running in every computer joining to the blockchain network.

Geth can be run as light node, which does not require to download the whole blockchain, but for the purposes of RGCE running full nodes is suggested at least in the

case of running a small network. If creating a realistic large, more difficult, and slower blockchain some of the nodes may be ran as light nodes, meaning they use light sync. Light sync only downloads block headers, instead of the whole blockchain. In the actual Ethereum network the blockchain takes a lot of diskspace if ran as a full node, but light node only requires couple hundred MB of memory, and the syncing to the main net is more efficient and faster compared to full node.

To start the network one node is used as the bootnode, meaning it is used to start the blockchain and every other node connects to it as peers. Running a boot node makes monitoring of the blockchain and nodes far less complicated, as everything is connected to one central point. Parallel blockchain networks can be run as simply as using different network IDs and different RPC addresses and ports.

After the Geth is installed it is used from the terminal using the “Geth” command. Using Geth requires the genesis block to be written in .json format file. After the genesis file is written the Geth can be started for the first time.

Figure 3 Starting the Ethereum blockchain

Executing the command to start Geth starts the blockchain and it is up and running, but not ready for use. (See Figure 3.)

4.3 Creating users

User accounts are used for storing Eth and identifying nodes in the blockchain network. One node may have many accounts, for different purposes and nodes are not bound to a singular address. User addresses are unique 42-byte long hex strings and built by applying Keccak-256 hash function to user’s public key, which is automatically generated.

After starting the Geth for the first time, first action should be an account creation to be used as the coinbase. (See Figure 4.) Coinbase is used as a wallet for miner to save the Eth rewards gained from mining blocks. Coinbase is a normal account, meaning transactions can be made from and to it and it can be used for smart contracts. Without a coinbase mining is impossible, as every block successfully mined rewards Eth, and it has to go somewhere. Coinbase can also be imported from Metamask or any other wallet that supports exporting accounts. Importing account from Metamask allows the miner to see and use the rewarded Eth easily from Metamask directly. After the coinbase account is made, allocating Eth to it in the genesis block file is possible simply by including the account address under the *alloc* flag and removing the existing blockchain followed by re-initializing the genesis block. This is especially useful in case of cyber security training, as the participants of training can have an account ready, that has a set sum of Eth already allocated to it.

Guide for joining the blockchain from other nodes can be found in Appendix 4.

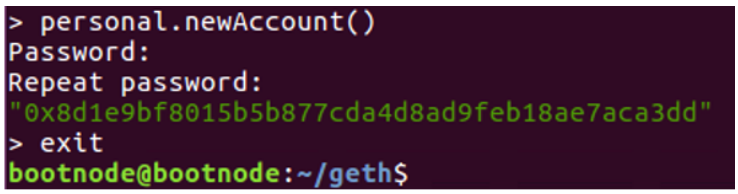

Figure 4 Creating a coinbase account in Geth

4.4 Metamask

Metamask is a browser extension that acts as an bridge between user and blockchain. It is used as a external wallet and making interactions with the Metamask is much more simplified compared to JS-command-line. Metamask runs as normal node and enables account creation, so end user does not have to use Geth JS-commands to create accounts and create transactions. Only downside of using Metamask in the test environment is that it can not be used for mining. However this has been taken into account and difficulty is set low in the blockchain, meaning only one node can be mining, and the blockchain still runs fast without any problems. Low difficulty means that there will be more blocks requiring more space, but in the case of private environment, the blockchain can always be completely removed after its

usage, if not more needed. Even though the blocks are generated fast, they do not contain as much data as a real block would in the main net, meaning the block size should not become a problem. Installation manual for Metamask can be found in Appendix 2.

After the coinbase account is made, no further accounts are needed to be made on the JS-command-line, but instead they are made using the Metamask.

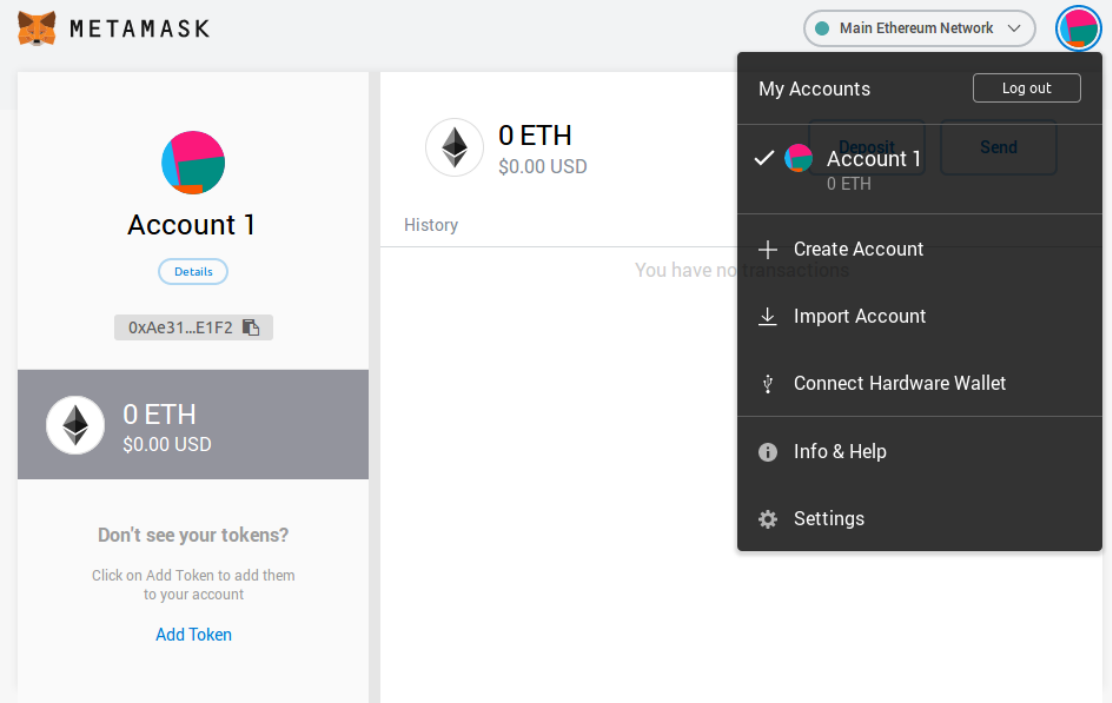

Figure 5 Metamask interface

Metamask interface is very simple and intuitive, thus not requiring practically any learning to use. (See Figure 5.) Metamask can create accounts and perform transactions. Accounts made in Metamask have 0 Eth just as in new accounts in the normal Ethereum blockchain would. Allocating Ether to Metamask accounts is possible, even though the blockchain needs to be removed, the Metamask account will still work. Removing the blockchain will remove the currency from both Metamask and Geth users.

4.5 Monitoring

Monitor is run from bootnode, and it can be run in localhost or it can be hosted to other networks. Monitor offers live time statistics of the blockchain running which is it connected to. Installation manual for monitoring is found in Appendix 3.

The monitoring is done using Eth-net-intelligence-api, which is the back-end part of monitor. It gets information from the blockchain by using JSON-RPC to get data straight from the node it is running on.

Eth-netstats is the front-end part of the monitor which gets information from eth-net-intelligence-api and transforms the data into a graphic form, which is easier to access and read. Eth-netstats gets data from eth-net-intelligence-api through the WebSockets protocol and outputs them through an angular interface.

Figure 6 Monitoring application accessed from the browser

The monitoring tool shows the current block, uncle blocks, last block and time elapsed from the mining of the latest block, average block mining time, average network hashrate, difficulty and nodes connected to the monitor. (See Figure 6.)
The monitoring tool is especially useful for analyzing the network, if blocks are mined slowly or the hashrate is dropping then something is going wrong or a miner might have stopped. Monitoring singular nodes is possible, making it easier to spot if nodes are not connected if they are supposed to be, or if nodes are experiencing problems with mining or interacting with the network.

4.6 Smart Contracts

Remix-IDE is an IDE for smart contract development. Remix-IDE is used on the machines that are using smart contracts. Remix-IDE uses Solidity as the language and it is able to connect to private networks. Normally Remix-IDE is used from browser connecting to the hosted service, but in RGCE environment a self hosted local version of Remix-IDE, which is connected to the private Geth network is used.

Implementing Remix-IDE in to RGCE allows development of smart contracts and DApps if they are needed for training or research purposes. To implement a smart contract a file needs to be made in the Remix-IDE file explorer which has ".sol" file-name extension. After smart contract is written it can be compiled and ran, as long as a machine is mining on the blockchain to lock the contract transactions to blocks. Running smart contracts also require small amount of Eth, to pay for gas fees.

To test that Remix-IDE is working properly, a simple "hello world" script was used for testing. After writing and compiling the smart contract it was deployed. After a contract is deployed it can be seen in the deployed contracts section, and by opening and rendering the deployed contract output of the contract a string "helloWorld" can be seen. (See Figure 7.) Installation manual for Remix-IDE can be found in Appendix 5.

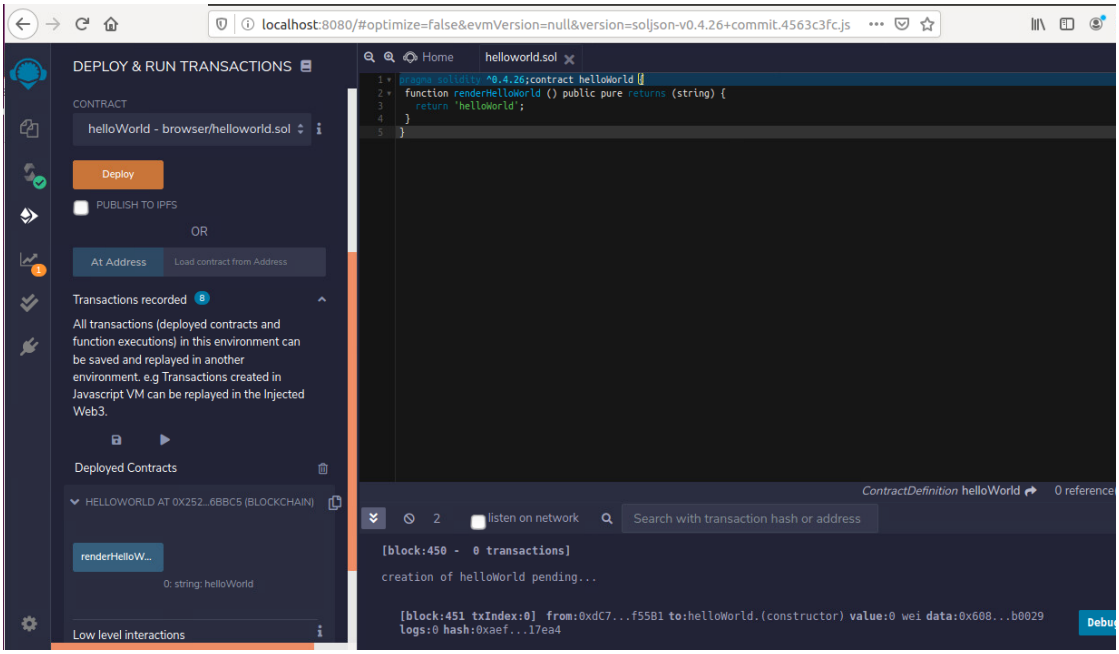

Figure 7 Hello world test

5 Attack vectors for blockchain and malicious use of blockchain

Blockchains are resistant to traditional attacks, but cybercriminals are developing new attacks and finding new attack vectors. Not only the blockchain network is at risk of attack, but so are the other parts built on top of the blockchain.

5.1 51% attack

Blockchain as a technology is considered very secure and hard to attack. The most probable attack on the blockchain network is the 51% attack. This type of attack is more likely to happen on smaller blockchain networks that do not have as much hash power as the major blockchains. If one entity manages to control over half of the network's hashrate, they can double spend. Double spending occurs, when a malicious entity who is performing the attack mines another chain forking from the correct chain. The attacker proceeds to stealth mine isolated chain while not broadcasting it to the rest of nodes connected to the main chain. Other nodes continue to mine and act with the blockchain, without knowing the malicious isolated chain is being mined. Attackers might be able to block some, or all transactions being confirmed, since on blockchain networks the majority has to approve transactions for them to happen. Blocking mining is also possible on some blockchains and it is known as mining monopoly.

The lifecycle of 51% attack. Attacker can spend all his currency on the actual chain knowing he will not actually lose any currency. The spending happens in the "Block 52" in this example. (See Figure 8.)

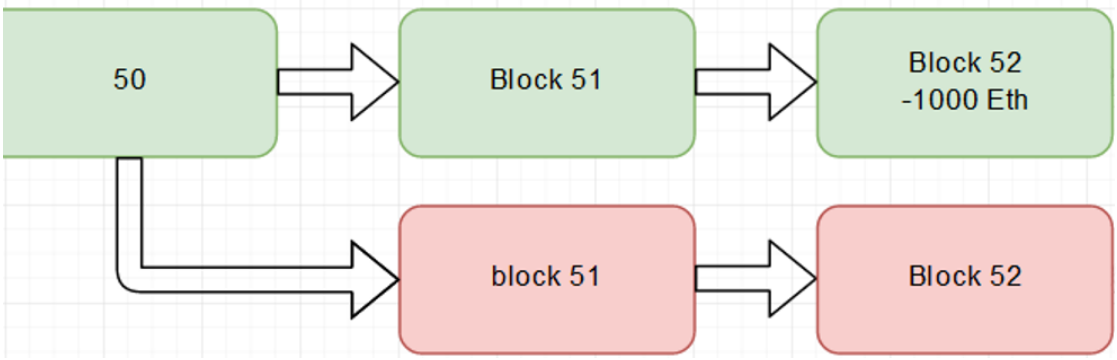

Figure 8 Stealth Mining

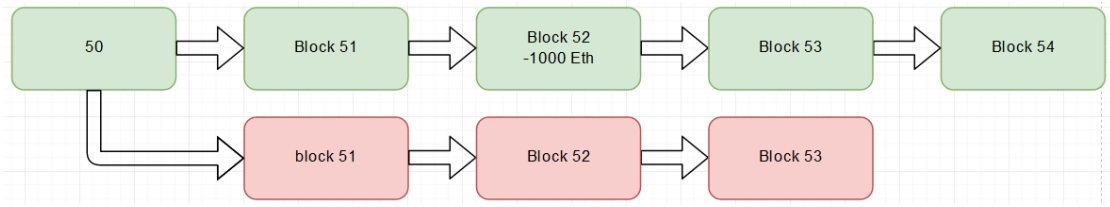

Figure 9 Continuing Stealth mining

This is where the attacker needs more hashing power compared to the rest of the network combined. (See Figure 9.) After the attacker's currency is spent, attacker then needs to race and finally overtake the original chain by mining ahead of the correct chain. (See Figure 10.)

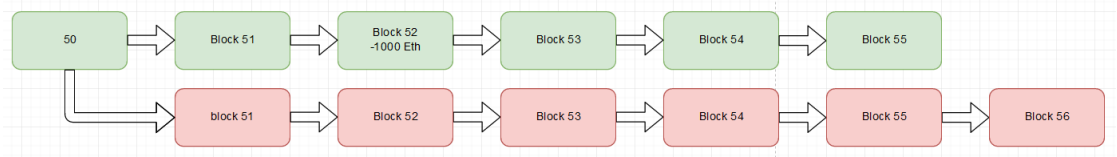

Figure 10 Racing the original chain

After the attacker catcher and gets ahead of the original chain the stealth mining is over and the fork will be broadcasted to the rest of the network. (See Figure 11.)

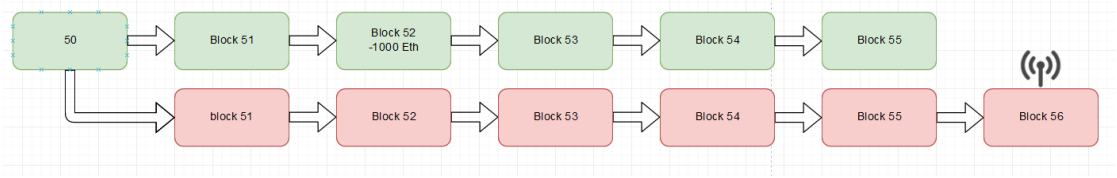

Figure 11 Broadcasting the malicious fork

Due to how blockchains work, the longest and "heaviest" chain is considered the right one. Rest of the network will join the corrupt chain and consider it the correct one. (See Figure 12.)

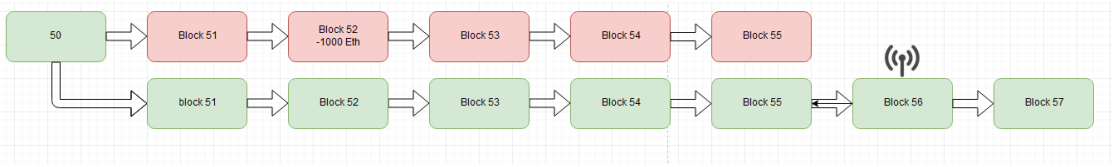

Figure 12 Network joining the corrupt chain

The corrupted blockchain is now considered the correct blockchain, and all the transactions that happened in the other chain after the stealth fork was made, will be reversed. Other people might lose or win when this happens, but reversing transactions is considered very dangerous since real money can be spent to buy currency, and suddenly they will not have the currency they bought. The attacker gains all the money spent on the previous chain back and the transaction made still stands in the old chain.

5.2 Sybil attack

Controlling multiple or a majority of the nodes on the network as a malicious actor enables manipulating the network. The attacker can outvote honest votes and even block transactions and even block other users from the network. In a large scale Sybil attack, the malicious actor might control over half of the hashing power of the network and perform a 51% attack.

A Sybil attack happens when one malicious actor controls a large number of the nodes in the network. Victim gets surrounded by attacking nodes and the attacker can then block transactions of the victim and possibly to double-spending attacks. (See Figure 13.)A Sybil attack is difficult to detect and prevent, but actions can be taken to prevent this type of attack happening. Using user power-based reputation or some other type of trust system for joining the network can be used as a solution to prevent the attack from happening

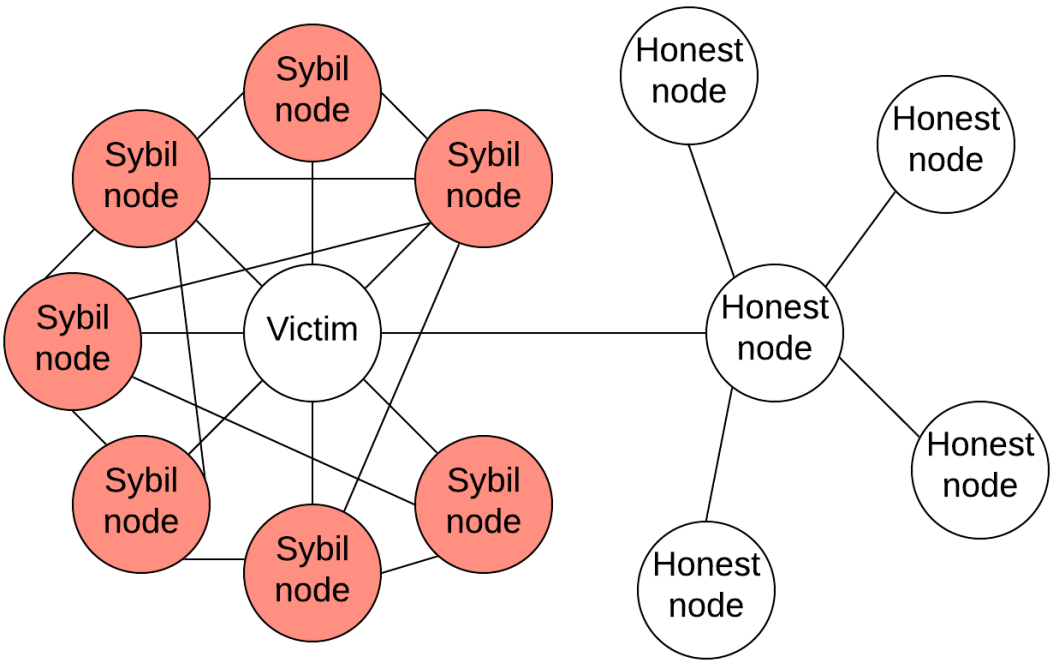

Figure 13 Sybil attack illustrated

5.3 DDoS

A Distributed Denial of Service attack is a common attack on centralized networks, but blockchains are effectively resistant to DDoS. Theoretically a blockchain network can be DDoS'ed, but in reality that would take such an immense amount of data and attackers, that it can be considered not a threat. Blockchains are effective protecting against DDoS attacks, as the workload is distributed to all nodes in the network. In blockchain networks the nodes may be pooled which allows larger amounts of data to be processed if DDoS is attempted on the network. Resilience against DDoS depends on the hash rate and size of the network. In larger network it does not cause problems if couple of nodes get disconnected from the network due to an attack.

5.4 Eclipse attack

An Eclipse attack is kind of a man in the middle attack where the attacker gets between the victim and the blockchain network. Performing such an attack requires large amount of IP addresses or botnet to overwrite victim nodes tried table. Each node has a table containing 4096 IP addresses of nodes that it has not connected to

yet. When the victim node is restarted, connections made by the victim are redirected to the attackers IP addresses. The victim node will be unable to interact with the network because it is connected to malicious nodes. Ethereum has patched eclipse attack and Ethereum networks are not vulnerable to eclipse attacks anymore.

5.5 Non blockchain specific attacks

Attacks exist specifically against blockchain networks and other parts of it, but other common attacks against centralized networks and applications do still apply to some parts of the blockchain and the environment around it.

5.5.1 Phishing

In 2018 IOTA blockchain had its wallets attacked with phishing. Attackers used fake seed generator, that allowed them to steal ~4 million USD worth of IOTA from the victims that had fallen to phishing.

Other blockchain wallets are no exception and phishing campaigns can still happen. Phishing attacks have been gaining popularity and cryptowallets and coin exchanges are phished regularly.

In the May of 2019 hackers managed to steal 41 million USD worth of Bitcoin from the Binance coin exchange. Hackers used variety of attack vectors such as phishing and viruses. In this attack no user lost any Bitcoin, as Binance had implemented a Secure Asset Fund for Users, for specifically these kinds of situations.

5.5.2 Dictionary Attacks

Passwords are only as strong as the user sets them. Weak and common passwords are being used and attackers are taking advantage of this. Wallets, computer, coin exchanges and even blockchain accounts all have passwords that are commonly stored in form of cryptographic hash. Attackers attempt to use common passwords and hash them to break the victims' hash and salt.

5.6 Smart Contract Attacks

DApps and smart contracts running on blockchain, are not nearly as secure as the blockchain itself. Smart contracts are essentially code powered by blockchain, making it as vulnerable as normal code. Coders of smart contracts need to take security into account when developing smart contracts and DApps. Transactions to DApps are ran through the blockchain network. Transactions are first relayed around the blockchain network until a miner puts the transaction in to a block (Eskendari, Mosaavi & Clark, 1).

For the scripting nature of Solidity, which the smart contracts are written in, many less severe vulnerabilities are discovered constantly.

5.6.1 Reentrancy

Reentrancy attacks have the greatest potential to cause harm. They can drain Eth out of smart contracts or even break the code. Reentrancy attack is possible when a smart contract contains function call to an external contract for its execution. If the malicious actor controls the external contract, they can repeat interactions that should not happen after the initial execution. For example a contract that does accounting, might expose a variable that shows how much balance the contract holds. If the contract doesn't set its balance to zero before sending Eth to recipient, the malicious recipient can write code that has fallback which calls withdraw repeatedly before the contract sets its balance to zero causing the contract to lose all balance it is holding.

5.6.2 Front-running

Front-running is an attack where a malicious node observes a transaction after it is broadcast but before it is finalized, and attempts to have its own transaction confirmed before or instead of the observed transaction (Eskendari, Mosaavi & Clark, 1). All transactions are visible in the network for a short period of time before they are locked into a block, which creates an opportunity of malicious actions. There are three kinds of front-running attacks.

Displacement is an attack, where the malicious actor sees transaction and proceeds to do the same transaction but with higher gas price. Attacker could see someone buying something and getting to buy it before the original buyer's transaction goes through.

An insertion attack is exactly how it sounds. Someone places a transaction on an asset at a higher than asking price. A malicious actor sees the transaction and buys the asset for the asking price and then inserts the same asset as an offer for the higher price. Then the original buyer gets the asset, but the malicious actor has made a profit. This is done by outbidding the original buyers gas price.

Suppression attack also known as a block stuffing attack happens when a malicious actor delays the transaction from happening. The malicious actor can increase the gas price and gas limit, to use all the available gas limit, thus delaying the original transaction from happening.

Front running attacks are caused by bad smart contract code and they can be solved by following good known practices.

5.6.3 Integer Overflow and Underflow

Solidity has integer types that have set maximum values, such as "uint8" = 255, "uint161" = 65535. Integer overflow and underflow may occur when the maximum or minimum values are exceeded. OpenZeppelin SafeMath library is suggested to be used when performing mathematical operations in smart contracts. The library is used to check for overflows.

5.6.4 DoS with block gas limit

Ethereum blocks have a gas limit, and if the gas limit of a block is exceeded, the transaction will not be accepted and will fail. This opens the way for attackers to attack in two different ways.

Unbounded operations may happen intentionally or by accident when creating transaction to an array of recipients. The gas limit might be exceeded without even trying. Malicious actor can take advantage of this by creating a massive array of recipients and sending minimal amounts of Eth using a smart contract, which can cause the

smart contract to not be able to process any other transactions. Attacker can also use multiple addresses to make small transactions using the smart contract. If the smart contract has a fallback function that refunds the Eth sent to it, but the gas cost of refund exceeds the gas limit the transaction will fail causing a DoS. To prevent this from happening each payment could be separated to individual transactions in the code.

Block stuffing DoS is more targeted attack as it is more effective at blocking transactions. Attacker can prevent other transactions by placing computationally intensive transactions with high gas price. Making multiple of transactions consuming the gas limit with high gas price will be included in the next block before the other transactions go through. This attack can be effective on time sensitive actions, when something needs to be done at specific time to get an advantage, the attacker can make the action and block all subsequent transactions. It is not 100% certain the attacker as no gas price can guarantee the next block for the intended transaction, but higher gas price increases chances of succeeding. This attack is only profitable when the profits of a successful attack exceeds the cost of the attack.

5.7 Ransomware

Ransomware criminals encrypt their victim's data, making their systems useless and data unavailable. The attackers are demanding a ransom, usually a sum of Bitcoin or other cryptocurrency. Often ransomware attackers use blockchain in ransomware attacks, as the payment method for decrypting data. Criminals like to use cryptocurrencies as it is easy to write software that asks for users to send cryptocurrency and in some cases performs the decryption of data when amount of currency is sent to the address.

In 2018, the HC7 Planetary ransomware was the first ransomware attack, where the attacker accepted Eth as payment for decrypting data. In addition to Ethereum the ransomware also accepted Bitcoin and Monero.

6 Hardening and mitigations

Blockchain and Ethereum are both considered safe in the eyes of the public, but malicious actors are finding more ways to break different parts of blockchain. Simply by following general good practices for cyber security such as keeping Go-Ethereum up to date, many of the known vulnerabilities can be prevented. Some safety precautions are in order when setting up an Ethereum network, especially if the blockchain is open to join from the outside of LAN.

6.1 Common Practices

Easy steps of hardening include very basic steps including using a strong password for accounts as well as using a firewall like UFW. The next step would be using reverse proxy such as Nginx and using HTTP basic auth.

RPC is not password protected for users and using a HTTP basic Auth will reduce the chances of RPC getting attacked by adding an additional layer of security by requiring username and password to access RPC. With a reverse proxy, IP whitelisting is a great way to prevent RPC access from unwanted IP spaces.

People are considered be the weakest link in the security. Human error and bad practices are the most common attack vectors. Phishing attacks have happened to Ethereum wallets and the best way to prevent phishing and other tricks to get access, is increasing knowledge and to train people. Good practice is to keep people well educated about secure computing practices, such as strong passwords, protecting computer, not installing any files from Internet and not leaving computer open or accessible.

6.2 51% attack

Fully protecting a blockchain from 51% attack is a nearly impossibile but making the attack far less feasible can be achieved. Efforts are made to get rid of the attack, but it is common in blockchains. Most promising of 51% attack mitigations are a form of increasing the cost of an attack so much that it won’t be profitable. As the attacker

would need to invest so much resources and money into the attack the attacker would essentially invest in the blockchain. Attacking the blockchain could crash the value of the currency of the blockchain and this effectively turns the attack cause a loss instead of a profit.

Researchers at the University of Luxembourg are part of an international team that has proposed the first blockchain system to guarantee proper performance even when more than 51% of the system's computing power is controlled by an attacker. The proposed idea is a reputation system that requires long term commitment of honest mining over long periods of time, effectively making 51% attack so expensive for the attacker that the attack would end up with loss instead of profit.

When Ethereum 2.0 will be launched, the 51% attack will be unlikely to happen anymore. 51% attacking a PoS network is more disadvantageous for the attacker, because the attack would need to require 51% of the currency to be held by the attacker. By holding 51% of the currency in the network the attacker would harm himself by attacking against the network he is the major shareholder of. If attacker successfully performs an 51% attack on PoS network, the attacker himself would suffer the biggest as the value of the currency would crash down after a network wide attack, thus making the currency held by the attacker lower in value.

Horizen has proposed a modified Satoshi consensus that implements a system that punishes for mining forks and delays the publishing of that fork. Implementing a delay to accept forks, can mitigate a 51% attack by requiring the attacker to mine much more, which causes the attack to not be economically profitable anymore. Penalty system of Horizen works following: Attacker disconnect from main net and starts to mine a fork. If attacker starts mining from block 100 and mines to block 119 but the main net is only at block 116, trying to publish the fraudulent fork as main chain, the system gives attacker penalty points for each block overwritten. In this case the attacker is trying to overwrite 16 blocks from 100 to 116, the system gives 16 penalty points. Attacker then must mine 16 blocks to get 1 penalty point off, followed by 15 blocks and so on. In total attacker needs to mine 133 blocks on top of the original 16 blocks he has already mined.

Using Bitcoin as an example for Horizen penalty system, the average block time is 10 minutes. This means that an attacker with 51% of calculating power takes ~20 minutes to mine a block and mining 133 blocks would take 2660 minutes, which is 880 hours meaning over a month of mining. Mining this long with the required amount of power would cost an immense amounts and the profitability would be non-existent.

6.3 Smart Contracts

Ethereum and blockchain programs are still quite new and experimental. New vulnerabilities known to public are patched by Solidity, but the major part of security lies in the hands of the writer of smart contract.

Consensys has written extensively on the topic of Ethereum smart contract security and best practices on their website (Consensys). General philosophy contains information about the mindset required to write secure smart contracts, general tips and what to expect when writing smart contracts. Smart contract recommendations demonstrate secure patterns of smart contracts that should be used when writing smart contracts. Best way securing smart contracts is gaining knowledge about possible attacks, attack mitigations and following the best practices.

7 Future development

7.1 Moving to PoS

When Ethereum 2.0 is published, the old Ethereum blockchain should be replaced by new PoS version. PoS requires less mining power, which makes running it in bigger scale easier. PoS increases the security of the blockchain network and makes 51% far less profitable for attackers.

7.2 Quantum computing

When quantum computers become a reality, they might be able to break public key encryption (Chattopadhyay 2017). While quantum computers are coming in the future, they are not stable enough yet and it might be over a decade before they become a reality concerning everyday computing. If blockchain continues to rely on public-key cryptography, quantum computing will eventually break the blockchain.

Quantum resilience should be a consideration for developers of blockchains. NSA has revealed it is working on “quantum-resistant crypto” and Andreas Antonopoulos who is a Bitcoin advocate has said that we should be ready for quantum computing when it breaks the elliptic curve.

8 Conclusions

While blockchain is still relatively new and upcoming technology, the hype around it has calmed down. While the hype is not as big as it used to be, blockchain is still in fast development, maybe waiting for something great to happen or to be developed for the breakthrough to the public. More blockchains are developed constantly all of which providing something the one before has not provided yet. Increased levels of building on top of the blockchain causes increased amount of attack vectors and this needs to be taken into account when using and developing them, especially in business.

Blockchain has been implemented in various ways into many different applications, but it has not found a practical use that will be widely used. Only when blockchain breakthrough happens and someone figures out an exceptional way to use it or how to merge blockchain into some other technology can we know the limits and possibilities of blockchain. It is impossible to predict the future, but the blockchain poses a great promise and possibilities.

Maybe the greatest potential of blockchain is in the security of the blockchain. While blockchain is considered very safe and nonpermutable, parts of it are susceptible to attacks. Especially smart contracts have many ways to break the code and cause harm.

While a 51% attack is the greatest risk for blockchain networks, the future for development has shown great potential to make the risk of 51% attacks less probable. Even completely 51% attack resistant blockchains have been proposed, but by definition blockchains are susceptible to 51% attacks no matter PoW or PoS.

Ethereum is kept up to date and the developers are actively making it more secure technology in all areas. The greatest concern of security lies in the DApps and smart contracts. Developing DApps or using smart contracts are still in the early stages and all the attack vectors might not have surfaced yet. Just as in normal app development security must be a major consideration from the start to the production.

Blockchains are still mostly used for the cryptocurrencies. An average user uses wallet to interact with the blockchain or coinexchange. For this kind of user, a strong password or buying a hardware wallet are the only options for securing their cryptocurrencies.

References

Asgaonkar, A. 2018. *Casper CBC, Simplified!.* Accessed on 10.12.2019 https://medium.com/@aditya.asgaonkar/casper-cbc-simplified-2370922f9aa6

Binance Academy. n.d. Byzantine fault tolerance Explained. Accessed on 11.10.2019 https://www.binance.vision/blockchain/byzantine-fault-tolerance-explained

Carlazo, L. 2017. What is Blockchain? Accessed on 8.6.2019. https://www.journalofaccountancy.com/issues/2017/jul/what-is-blockchain.html

Chattopadhyay, A. 2017. IoT Security Using Blockchain. Accessed 3.5.2019 https://www.uk.sogeti.com/content-hub/blog/iot-security-using-blockchain/

Compton, J. 2017. Is Blockchain the Solution to Iot Security? Accessed on 13.3.2020. https://innovationatwork.ieee.org/blockchain-iot-security/

Consensys. n.d. Smart Contract Best Practices. Accessed on 19.2.2020 https://consensys.github.io/smart-contract-best-practices/

Dongxing, L. Wei, P. Wenping, D. Fangyu, G. 2018. A Blockchain-based Authentication and Security Mechanism for IoT. Accessed on 13.3.2020. https://www.researchgate.net/publication/328247073_A_Blockchain-Based_Authentication_and_Security_Mechanism_for_IoT

Eskendari, S. Mosaavi, S. Clark, J. 2019. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. https://users.encs.concordia.ca/~clark/papers/2019_wtsc_front.pdf

Ethereum n.d. Accessed on 18.9.2019. https://Ethereum.org/

Ethereum GitHub. Ethash. n.d. Accessed on 8.3.2020 https://github.com/ethereum/wiki/wiki/Mining#ethash-dag

Etherscan. n.d. Accessed 16.9.2019. https://etherscan.io/blocks

EthHub. History and Network Upgrades. 2015. Accessed on 30.9.2019 https://docs.ethhub.io/Ethereum-basics/history-and-forks/

Haber, S. Stornetta,S. How to Time-Stamp a Digital Document. Accessed on 22.5.2020. https://www.anf.es/pdf/Haber_Stornetta.pdf

Miles, C. 2017. *Blockchain security: what keeps your transaction data safe? Accessed on 22.8.2019.* https://www.ibm.com/blogs/blockchain/2017/12/blockchain-security-what-keeps-your-transaction-data-safe/

Praitheeshan, P. Pan, L. Yu, J. Robin, D. 2020. Security Analysis Methods on Ethereum Smart Contract Vulnerabilities – A Survey. Accessed on 14.3.2020. https://arxiv.org/pdf/1908.08605.pdf

Solidity. n.d. Accessed on 12.1.2020. https://solidity.readthedocs.io/en/v0.6.4/

Zainuddin, A. n.d. Guide to Ethereum: What is gas, gas limit and gas price? Accessed on 16.9.2019. https://masterthecrypto.com/Ethereum-what-is-gas-gas-limit-gas-price/

Appendices

Appendix 1. Installation manual for Go-Ethereum

Geth is run on Ubuntu 16.04, but other Linux distributions and Windows machine can be used.

A directory must be created for blockchain datadirectory. Installing Geth can be done using following commands:

- sudo apt-get install software-properties-common
- sudo add-apt-repository -y ppa:ethereum/ethereum
- sudo apt-get update
- sudo apt-get install ethereum -y

After successfully installing Geth it can be started for making a coinbase account, which is required for mining. For starting of Geth following command is used:

- sudo geth console 2>log.log

After Geth is started a JS console is used to create an account, which will be automatically used as the coinbase. Coinbase is set and Geth can be closed using following commands:

- personal.newAccount()
- Exit

Geth has been started once and before using it a configuration file for genesis block needs to be made. Coinbase account can be used in the genesis block for allocating Ether. (See Figure 2.) Genesis block needs to be initialized, but as the blockchain has started once, a database has been established and it needs to be removed, before the genesis block can be initialized by using following commands:

- sudo geth –nousb –datadir /pathname/directory_used_for_go-ethereum removedb
- sudo geth –nousb init /pathname_where_genesis_file_is_located/genesis.json –datadir /pathname/datadirectory_for_go-ethereum

Now Geth can be started properly. (See Figure 3.) Starting go-ethereum is done using following command:

- sudo geth –identity “name” –datadir /pathname/directory_of_go-ethereum –networkid 123 –nousb –rpc –rpcport “8545” –rpcaddr “localhost” –rpccorsdomain “*” –rpcapi “db, eth, net, web3, miner” console 2>log.log

Starting commands explained:

- *Identity* is the custom name for the node.
- *Datadir* points to the directory for databases and keystores.
- *NetworkID* is custom integer which is used to identify network.
- *NoUSB* disables monitoring for usb wallets. If not used, may prompt error when run.
- *RPC* enables HTTP-RPC server. RPC is used for external tools to manage and observe the blockchain.
- *RPCport* defines the port used for RPC, default port is 8545.
- *RPCaddr* points to RPC listening interface, default localhost.
- *RPCcorsdomain* list of domains to allow cross origin requests, “*” means allow all.
- *RPCapi* list of API’s used for RPC.
- *Console* starts a JS console.
- *2>log.log* saves the output of the Geth to log.log file.

Blockchain will start and if machine is a mining machine the mining can be started using the command:

- miner.start(number_of_threads_used_for_mining)

Difficulty of the blockchain should be set low in the blockchain file’s “difficulty” parameter, so only one machine can do the mining work for the whole blockchain. Mining will not start immediately as a DAG must be generated, which can take couple of minutes. To follow the progression of DAG generation and mining, “tail” can be used to view the log file in real time:

- tail -f “logfile.extension”

Appendix 2. Installation manual for Metamask

Metamask is installed from the browser extension store. After Metamask is installed to the browser, an account is automatically created, and additional accounts can be created simply by using the wallet interface's "create account".

Metamask is connected to the private blockchain, by using the network selection in the extension. (See Figure 14.) If blockchain is run on localhost and in default RPC-port the "Localhost 8545" can be selected and Metamask will connect to the block-chain. If the blockchain is hosted to a network and is using different RPC-port, the "Custom RPC" selection enables the configuration for custom address and port used. (See Figure 15.)

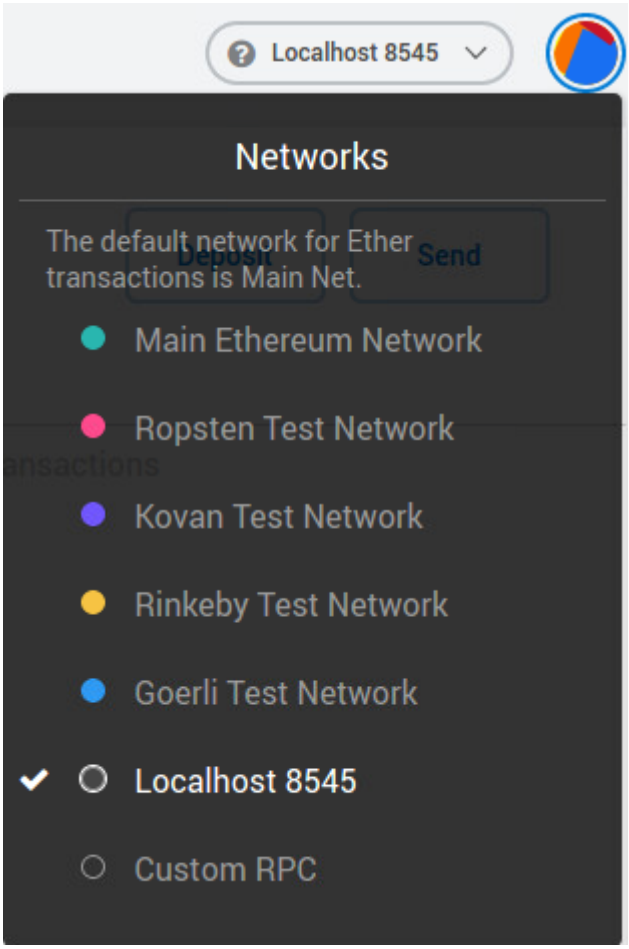

Figure 14 Metamask network selection

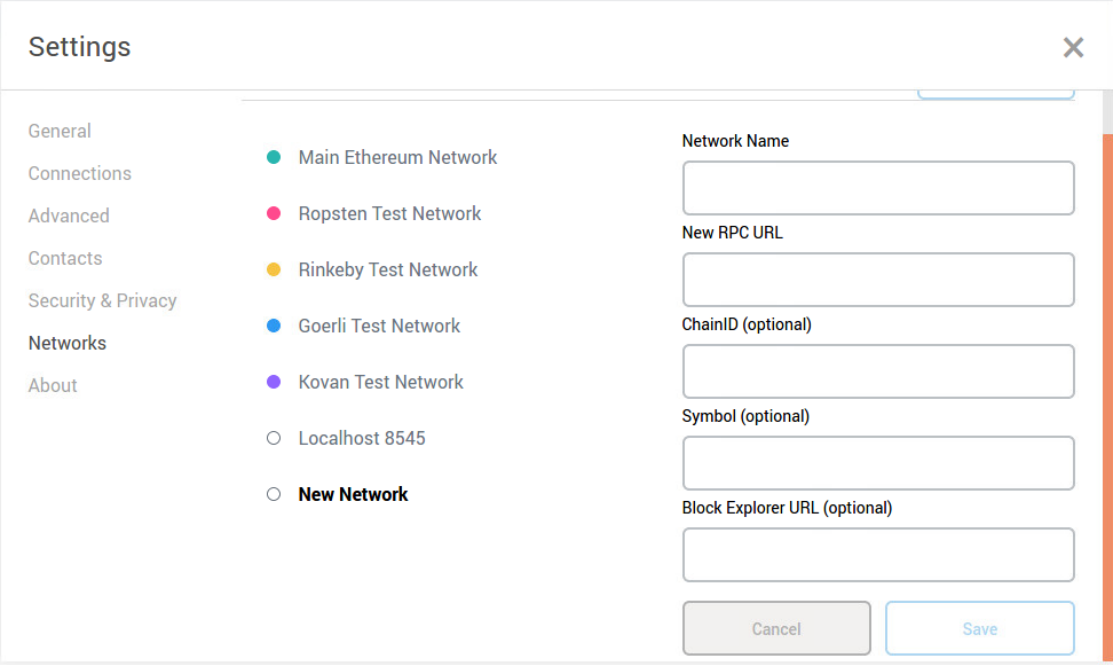

Figure 15 Custom RPC configuration

To get Ether on Metamask accounts, a transaction can be created from Geth JS console by using command:

- Eth.sendTransaction({from:sender_address, to:receiver_address, value: web3.toWei(amount, “ether”)})

Another way to get Ether on Metamask account is to allocate Ether to it in the genesis block file. Metamask account can also be used as a coinbase account and Ether rewarded from mining will go directly to Metamask by using following command in Geth JS console:

- miner.setEtherbase(“account_address”)

Appendix 3. Installation manual for blockchain monitoring

#Prerequisites

Eth-net-intelligence-api and eth-netstats both require following software to be installed:

- Git
- Curl
- NodeJS

NodeJS requires Curl to be installed on the machine, you can install NodeJS using following commands:

- sudo curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash –
- sudo apt-get install -y nodejs

Installing eth-net-intelligence api

Now that node is installed, the PM2 which is used to run the eth-net-intelligence-api can be installed and the repository for eth-net-intelligence-api can be downloaded using following commands:

- sudo npm install -g pm2
- sudo git clone https://github.com/ethereum/eth-net-intelligence-api

Configuration file “app.json” is found in the eth-net-intelligence-api directory. In the configuration file params “name”, “INSTANCE_NAME”, “WS_SERVER” and the “WS_SECRET” need to be configured before using. (See Figure 16.) Parameters “name” and “INSTANCE_NAME” are identifiers used for the monitoring. “WS_SERVER” needs to point to eth-netstats server. If running in localhost “localhost:3000” works, but if hosted to a network, then an address needs to be given. “WS_SECRET” is a password used for allowing eth-netstats to connect to eth-net-intelligence-api.

Each machine joining the monitoring needs its own configuration file, but in the nodes connecting to the monitoring node the “RPC_HOST” needs to be the IP-address of the node joining the monitoring.

Figure 16 app.json configuration file

After configurations have been set, the eth-net-intelligence-api can be started by using command:

- sudo pm2 start app.json

If other machines are used in the monitoring, then every configuration file must be included in the starting command:

- sudo pm2 start app.json app1.json app2.json

#Installing eth-netstats

For eth-netstats Grunt is installed and it is used for building the resources for the monitor interface. Installing eth-netstats and Grunt is done by using following commands:

- sudo git clone https://github.com/cubedro/eth-netstats
- cd eth-netstats
- sudo npm install
- sudo npm install -g grunt-cli
- Grunt

To start the eth-netstats the “WS_SECRET” set in the eth-net-intelligence api is used in the starting command:

- Sudo WS_SECRET=password npm start

After executing start command, the monitor is accessible from the browser at the set address. (See Figure 6.)

Appendix 4. Joining the blockchain

On the nodes joining the blockchain an exact copy of the genesis block have to be initialized and in the starting command of Geth “id” has to match the blockchain network id and the “rpc_addr” needs to be set to IP-address of the joining node. After the blockchain is started from the joining node, the node must be added as a peer in the node that is being joined to. To add peers, an enode address is required. Enode address of the node can be found from the Geth JS console by using command:

- admin.nodeInfo

In the end of the enode address is an IP-address and port. The IP address needs to match with the IP-address of the joining node.

After the IP address at the end of the enode address correctly, the peer can be added by using command:

- admin.addPeer(“enodeaddress@IP:port”)

If peer is added successfully the console should return “true” and to verify that peer has been successfully added peers can be viewed with a command:

- admin.peers

After the node joins the blockchain it starts syncing blocks, this can take some time anywhere from seconds to tens of minutes, depending on the size of the blockchain. To check that syncing is done, on both machines the highest block can be seen with command:

- eth.blockNumber

If the returned number match, then the syncing is completed, and mining can be done from the peer and the block number will go up on both machines.

Appendix 5. Installation manual for Remix-IDE

Remix-IDE requires older version of NodeJS and the version 13.x does not work. Version 10.x needs to be used. To see how to install NodeJS, See Appendix 3. Deploying smart contracts require an unlocked account and "*—allow-insecure-unlock"* needs to be included in the Geth starting command to be able to unlock accounts. In addition to NodeJS, Remix-IDE requires installation of:

- build-essential
- node-gyp
- apache2

These can be installed with commands:

- sudo apt-get install -y build-essential
- sudo npm install -g node-gyp
- sudo apt-get install apache2

Remix-IDE can be installed with command:

- sudo npm install -g remix-ide –unsafe-perm=true –allow-root

Remix-IDE is started by using command:

- remix-ide

Remix-IDE starts, and it can be accessed from the browser by going to the address, which can be seen in the console after Remix-IDE has been started.

To start using the Remix-IDE an environment needs to be chosen. Solidity is the main environment for smart contracts, and it needs to be chosen in the starting page.

After the environment is chosen, the Remix-IDE needs to connect to the blockchain, which can be done from Deploy & Run Transactions section. (See Figure 17.) Choosing Web3 provider enables the connecting to a private blockchain. After connecting Remix-IDE to the private blockchain, a smart contracts can be deployed.

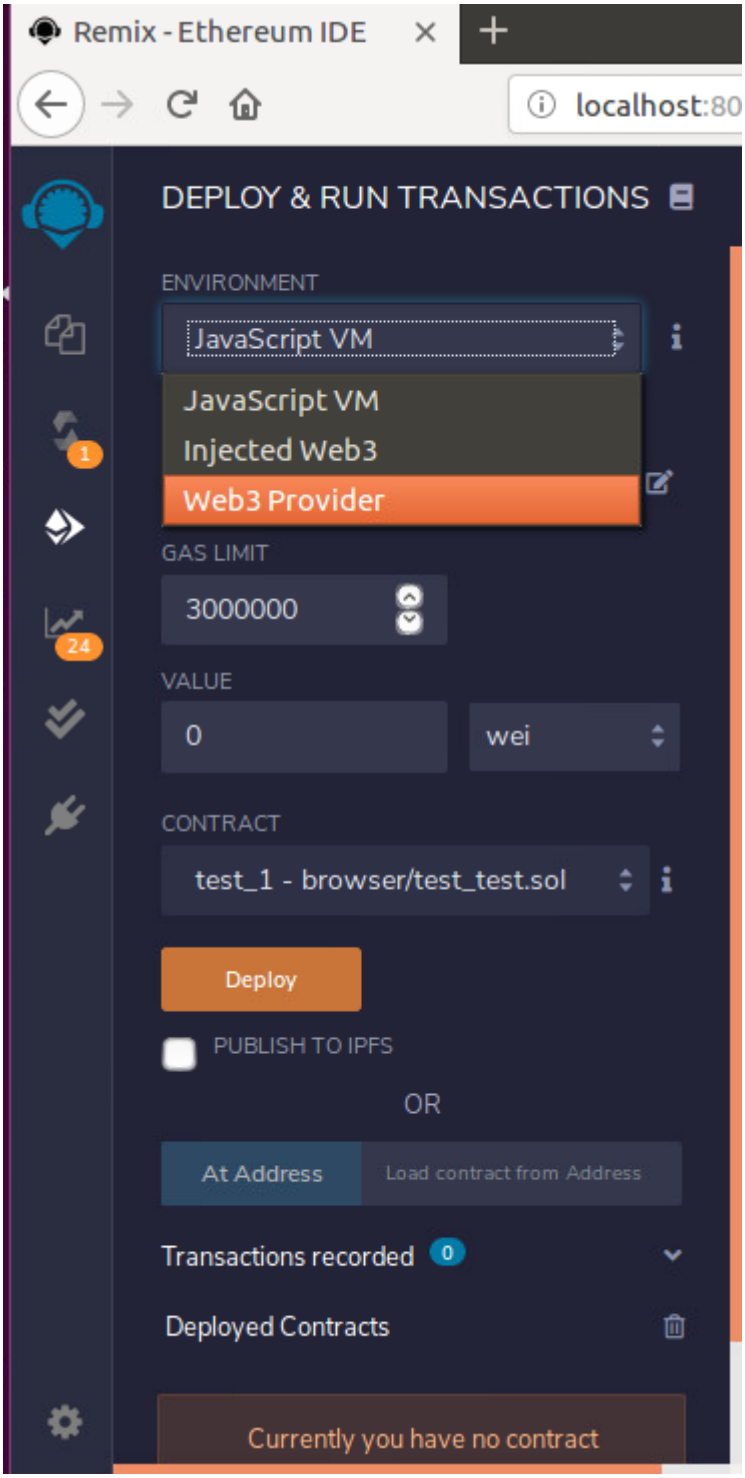

Figure 17 Choosing the blockchain environment

Smart contracts can be developed in the file explorers section by creating a new file with .sol file extension. After contract is written it needs to be compiled in the Solidity compiler section. After compiler has finished compiling the contract, a green check mark will appear on the icon of Solidity compiler.

Deploy & run transactions section is used for deploying smart contracts. To deploy a smart contract an account must be selected, because deploying a smart contract requires an account to pay for the gas fees, as the smart contract is a transaction. Account used for the smart contract deployment needs to be unlocked from Geth JS console using command:

- personal.unlockAccount(eth.accounts[x], “password”)

After account is unlocked the smart contract can be deployed. After a block is mined where the smart contract is included, it will be shown in the deployed contracts section. By pressing “Render” anything returned from the smart contract can be seen. (See Figure 7.)